Compte rendu de groupe

Groupe __	
Elève 1	COTTARD Arthur
Elève 2	BLANC Thomas
Elève 3	DUGUÉ Maxime
Elève 4	CHEREL Côme

CONSIGNES :

Ce compte rendu est à compléter ci-dessous au sein de chaque groupe.

1. Pour chaque tâche individuelle :
   - L'auteur de la tâche rédige ici son compte rendu en mentionnant d'abord son nom puis en fournissant exclusivement le « bout » de code qu'il a ajouté. Il n'est demandé qu'une fonction par élève parmi celles identifiées **F09** (`fenetreBalcon`), **F11** (`rdc`), **F12** (`etage`) et **F14** (`immeuble`).
   - L'auteur explique ce qu'il y a de nouveau pour lui (par exemple pour le module trait, c'est turtle avec les différentes instructions, pour la couleur le randint …), quelques lignes suffisent,
1. L'auteur justifie si nécessaire la structure algorithmique qu'il a choisie : conditionnelle, répétitive.
2. Pour le groupe, formaliser une conclusion collective en argumentant les réponses :
   - Ce que vous a apporté le projet ou pas ;
   - Les difficultés rencontrées ou les facilités ;
   - L'expérience acquise ;
   - Les éventuels retours sur des notions que nous devons revoir en cours car mal compris ou au contraire les notions que vous avez sues réinvestir car vous les maitrisiez ;
   - Tout autre élément que vous voudriez porter à notre connaissance…

COMPTE RENDU :

N.B : Notre projet est divisé en deux parties, une première portant sur le projet original, avec le cahier des charges et les compte-rendus individuels, et une seconde partie sur le projet évolué, comportant des optimisations hors du cahier des charges ou des fonctionnalités plus poussées intégralement commentées.
On compte parmi ces fonctionnalités :

- Une mise en perspective cavalière de la rue.
- Des ajouts esthétiques tels que de la végétation et du mobilier urbain entre autres.
- Un cycle jour-nuit synchronisé sur l'heure réelle.

Chaque membre du groupe a contribué au deux versions du projet.
Vous trouverez le détail de cette version du projet en partie annexe et sur Github.

CÔME CHEREL (F12) :

```
# dessin des murs
  facade(x,ySol,couleur,niveau) # dessine la facade aux coordonnées x,ySol et au niveau et dans une couleur définis en argument
  # dessin des 3 Eléments
  x_elements = [x,x+42.5,x-42.5] # crée une liste contenant l'abscisse de chaque élément
  shuffle(x_elements) # mélange de façon aléatoire les éléments de la liste grâce à shuffle
  for i in range(3): # boucle qui se répetera 3, où i prendra successivement les valeurs 0, 1 et 2
    f_ou_fb = randint(0, 1) # initialise une variable contenant un entier aléatoire entre 0 et 1
    if f_ou_fb == 0: # structure conditionnelle afin de dessiner une fenetre avec ou sans balcon en fonction de la variable ci-dessus
      fenetre_balcon(x_elements[i],ySol+niveau*60) # dessine une fenetre avec un balcon aux coordonnées x,ySol et au niveau défini en argument
    else:
      fenetre(x_elements[i],ySol+20+niveau*60) # dessine une fenetre simple aux coordonnées x,ySol et au niveau défini en argument
```

Dans la fonction étage, j'ai découvert les listes et j'ai commencer à comprendre son fonctionnement. De plus, j'ai appris la signification de la fonction shuffle qui consiste à mélanger les éléments de la liste. Pour cette algorithme, j'ai choisi une structure répétitive avec le fonction (for i in range():).

MAXIME DUGUÉ (F09) :

```
turtle.fillcolor("light blue") # défini la couleur de remplissage sur bleu clair
  turtle.begin_fill() # commence le remplissage
  rectangle(x,y,30,50) # dessine la porte fenêtre aux coordonnées x,y de largeur 30 et de hauteur 50
  turtle.end_fill() # termine le remplissage

  # balcon
  rectangle(x,y,40,30) #trace le contour du balcon aux coordonnées x,y de largeur 40 et de hauteur 30
  for barreau in range(-3,4): # boucle pour tracer les barreaux du balcon
    trait(x+5*barreau,y,x+5*barreau,y+30)# trace chaque barreau dont l'abscisse varie en fonction du compteur de la boucle
```

Dans ce code j'ai utilisé une structure répétitive a la fin pour éviter de répéter les mêmes choses avec juste une modification.C'est ce qu'il y avait de nouveau pour moi et c'était utile pour rendre le code plus court et plus lisible.

THOMAS BLANC (F11):

```
# Dessine la facade
  facade(x, ySol, c_facade,0) # dessine la facade de couleur définie en argument aux coordonnées (x,ySol) au niveau 0

  # Construit les 3 éléments (1 porte et 2 fenetres)
  x_elements = [x,x+42.5,x-42.5] # liste contenant les différentes abscisses des éléments
  shuffle(x_elements) # mélange de façon aléatoire les élément de la liste
  fenetre(x_elements[0],ySol+20) # dessine une fenetre à l'abscisse du premier élément de la liste et à 20px du sol
  fenetre(x_elements[1],ySol+20) # dessine une fenetre à l'abscisse du deuxième élément de la liste et à 20px du sol
  porte(x_elements[2],ySol,c_porte) # dessine la porte à l'abscisse du troisième élément de la liste au niveau du sol dans une couleur definie en argument.
```

Pour moi ce qu'il y a de nouveau dans cette fonction rdc est la création d'une liste ou l'on rend l'ordre de ses éléments aléatoire grâce a la fonction shuffle pour que l'on ne sache pas ou vas apparaître sur la façade chaque éléments c'est a dire quelle sera l'abscisse de la fonction porte et des 2 fonctions fenêtres.

COTTARD ARTHUR (F14) :

```
# Nombre d'étage (aléatoire)

  nb_etage = randint(1,4) # Détermine de façon aléatoire le nombre d'étages et le stocke dans
une variable

  #Couleurs des éléments (aléatoire)
  turtle.colormode(255) # Défini le mode de couleur de turtle sur 255
  couleur_facade = couleurAleatoire() # Défini la couleur de la facade aléatoirement
  couleur_elements= couleurAleatoire() # Défini la couleur des éléments aléatoirement

  # Dessin du RDC

  rdc(x,ySol,couleur_facade,couleur_elements) # Dessine le rdc aux coordonnées x,ySol avec les
couleurs définies plus haut

  # Dessin des étages
  for niveau in range(1,nb_etage+1): # boucle qui s'exécutera en fonction du nombre d'étages
    etage(x,ySol,couleur_facade,niveau) # Dessine un étage aux coordonnées x,ySol avec la cou-
leur de la facade plus haut et le niveau

  # Dessin du toit
  toit(x,ySol,nb_etage+1) # Dessine un toit qui sera choisi aléatoirement
```

Cette fonction m'a permit de découvrir le fonctionnement de turtle, et notamment colormode() afin d'utiliser des valeurs rgb. J'ai utilisé une structure répétitive afin de réduire le nombre de ligne, et donc rendre le code plus lisible.

CONCLUSION COLLECTIVE :

Le projet a tout d'abord permit au groupe de se familiariser avec l'outil de travail Github, bien que les automatismes n'aient pas été acquis par tous, cet outil nous a rendu plus efficaces.

Le fonctionnement des modules tels que random avec randint et shuffle, ou turtle pour dessiner a été correctement compris par l'ensemble du groupe, avec parfois quelques difficultés pour certains à initier le raisonnement qui nous permettrait de trouver la solution aux problèmes posés à partir du cahier des charges, comme rendre la position des éléments aléatoire sur la façade.

L'utilisation de l'IDE PyCharm, permettant de tester directement le code a plu au groupe, notamment pour perfectionner son travail, bien que certaines notions du langage Python restent flous pour certains, comme les listes et les tuples. En revanche, les boucles ont été facilement mises en places.

Partie Annexe :

Fonctionnalités ajoutées :

- Mise en perspective cavalière de la rue :
  - Création d'une fonction parallélogramme, indispensable pour construire un pavé.
  - Création d'une fonction pavé semblable à rectangle mais prends en compte l'angle de fuite pour construire les faces visibles à l'aide de parallélogrammes.
  - Ajout de la possibilité de mettre en relief tous les éléments dit « complexes » avec une profondeur propre à chaque élément de façon réaliste
  - Seul l'angle de fuite fait varier la perspective, définir l'angle comme None renvoie la rue classique, en « 2D ».
- Ajout d'éléments esthétiques :
  - Végétation : arbres, buissons.
  - Mobilier urbain : lampadaires, clôtures.
  - Ciel : couleur de ciel, nuages, étoiles et astres
- Ajout d'un cycle jour-nuit basé sur l'heure réelle :
  - L'heure du système est récupérée et exploitée. Une heure « artificielle » peut être entrée.
  - La couleur du ciel varie en fonction du cycle.
  - Les étoiles apparaissent la nuit, les nuages le jour.
  - Les fenêtres, s'illuminent aléatoirement, les lampadaires s'allument systématiquement la nuit et s'éteignent le jour.
- Optimisations diverses :
  - Le remplissage s'effectue directement dans la fonction rectangle ou parallélogramme, avec une valeur passée en argument. Des bugs ont été constatés avec l'utilisation redondante de turtle.begin_fill…
  - La fonction pavé a remplacé la fonction rectangle partout où un élément nécessitait d'être mis en relief, afin d'obtenir tout de même un rectangle, on définit l'angle sur None.
  - Le paramètre tridi à été ajouté partout où un élément pouvait être mis en relief. Il correspond à l'angle de fuite de la perspective.

Lien Github : Github-Architecte